**CSC 4710  
DATABASE SYSTEMS**

**PROJECT:**

**NBA PLAYER DATABASE**

**Team Name: Something Short**

**Team Members: Stallone Eu**

**Victoria Green**

**Jarrod Vega**

**Date: 12/7/2016**

**Contents**

**Table of Contents…………………………………………………………...…………1**

**Original Project Idea……………………………………………………...…….…..…2**

**System Requirements………………………………………………………………...2**

**Contextual Data Flow Diagram……………………………………………………...3**

**ER Diagram……………………………………………………………………………..4**

EER Diagram……………………………………………………………………………4

**Normalization…………………………………………………………………………..4**

1NF………………………………………………………………………..………………5

2NF………………………………………………………………………………………..6

3NF………………………………………………………………………………………..7

**Recommendation on DMBS and Rationale………………………………….……8**

**Implementation-Ready Data Model………………………………………….……..8**

**Data Dictionary…………………………………………………………….…………..9**

**SQL……………………………………………………………………………………...11**

Useful Queries……………………………………………………….………………...16

**Time Log……………………………………………………………………………….19**

**Appendix……………………………………………………………………………….19**

**Original Project Proposal**

I’m looking at a database that stores a *sports league’s.* Its teams,players,venues,team stats, individual player stats. Each TEAM has a primary key id,attributes such as name,date formed,web\_site,zip code. PLAYER have a primary key player id, an attributes like name,date of birth,college id. VENUES primary key venue id, then zip code,name,etc. STATS players id,primary key stats id,game id,points scored,assist,fouls,steals,etc. Database will keep track of wins and losses by each team and who the team loses and wins too. Will also store players stats and history in college,or highschool if they played. This database can also have the flexibility to have more criteria added in each column to give a more comprehensive overview of the league as a hole.

**1. System Requirements**

The purpose of our NBA Player database is to allow NBA Coaches to store and update information about their players and teams. It will also allow NBA players and NBA fans to access and read the information in this database by querying.

1.1) Each player does not have to attend a college, but they can only attend one college

1.2) Each college can have one or many players

2.1) Each player is on exactly one team

2.2) Each team has at least 5 players, but can have many

3.1) Each college has exactly one location

3.2) Each location can have either no college, or many colleges

4.1) Each location can have either no team, or it can have many teams

4.2) Each team has exactly one location

5.1) Each team can be in only one conference (Eastern or Western)

5.2) Each conference must have at least one team, up to 15 teams

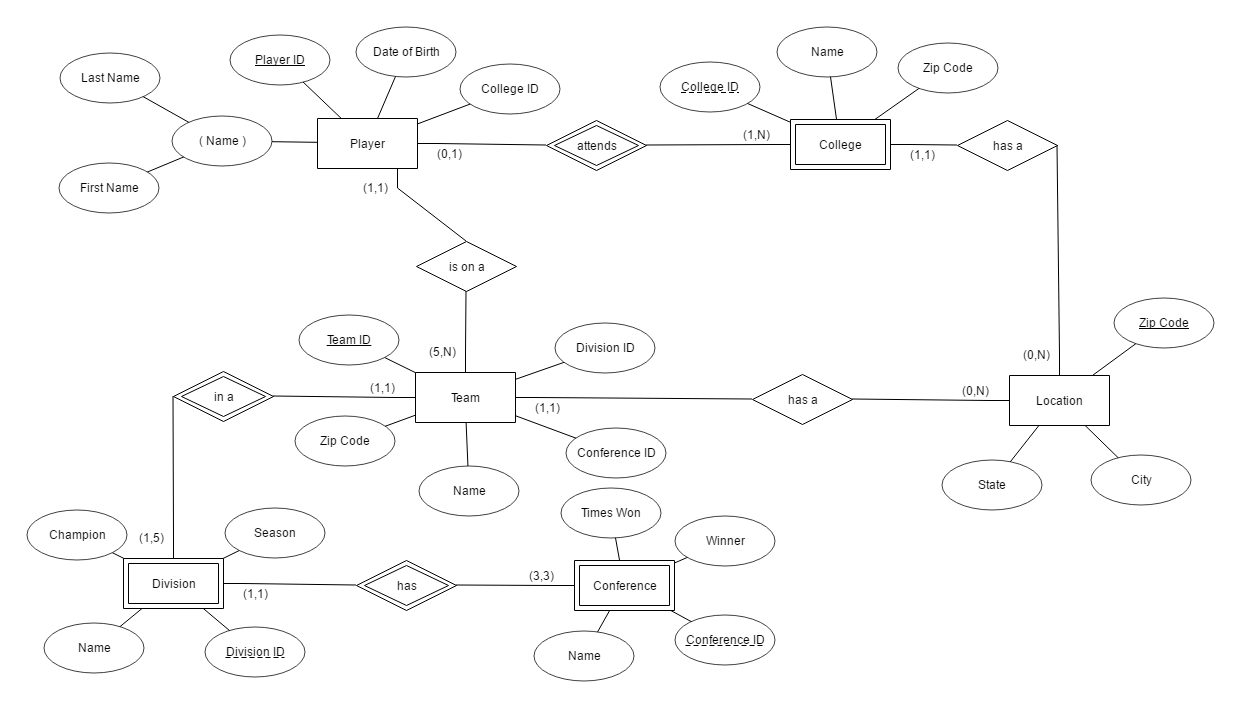
6.1) Each conference has exactly 3 divisions (Eastern conference has Atlantic, Central, and Southeast. Western conference has Southwest, Northwest, and Pacific)

6.2) Each division is in exactly one conference

**2. Contextual Data Flow Diagram**

**Contextual Data Flow Diagram.png**

**3. ER Diagram**



**EER Diagram**

We have decided that our database does not require an EER diagram. We believe our database does not include any set-subset relationships, or any specialization/generalization hierarchies.

**4. Normalization**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Player** |  |  |  |  |
| Player ID | College ID | First Name | Last Name | Date of Birth |
|  |  |  |  |  |
| **Team** |  |  |  |  |
| Team ID | Division ID | Conference ID | Zip Code | Name |
|  |  |  |  |  |
| **Location** |  |  |  |  |
| Zip Code | State | City |  |  |
|  |  |  |  |  |
| **College** |  |  |  |  |
| Player ID | College ID | Zip Code | Name |  |
|  |  |  |  |  |
| **Conference** |  |  |  |  |
| Conference ID | Team ID | Name | Winner | Times Won |
|  |  |  |  |  |
| **Division** |  |  |  |  |
| Division ID | Team ID | Name | Season | Champion |

**1NF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Player** |  |  |  |  |
| Player ID | College ID | First Name | Last Name | Date of Birth |
|  |  |  |  |  |
| **Team** |  |  |  |  |
| Team ID | Division ID | Conference ID | Zip Code | Name |
|  |  |  |  |  |
| **Location** |  |  |  |  |
| Zip Code | State | City |  |  |
|  |  |  |  |  |
| **College** |  |  |  |  |
| Player ID | College ID | Zip Code | Name |  |
|  |  |  |  |  |
| **Conference** |  |  |  |  |
| Conference ID | Team ID | Name | Winner | Times Won |
|  |  |  |  |  |
| **Division** |  |  |  |  |
| Division ID | Team ID | Name | Season | Champion |

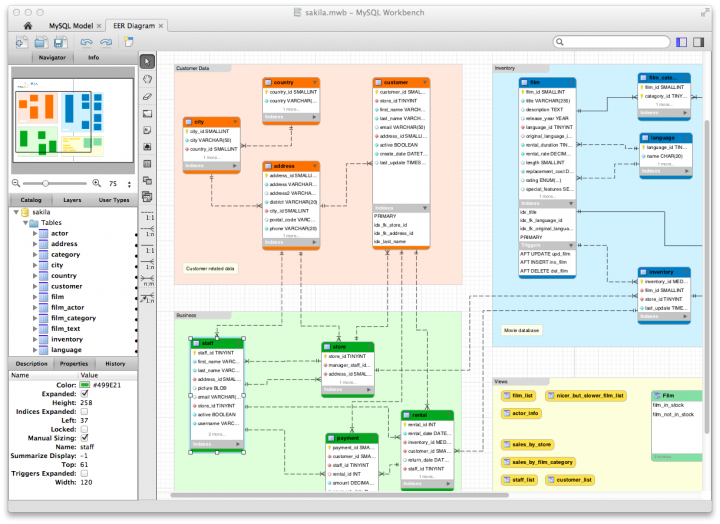
**2NF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Player** |  |  |  |  |
| Player ID | College ID | First Name | Last Name | Date of Birth |
|  |  |  |  |  |
| **Team** |  |  |  |  |
| Team ID | Division ID | Conference ID | Zip Code | Name |
|  |  |  |  |  |
| **Location** |  |  |  |  |
| Zip Code | State | City |  |  |
|  |  |  |  |  |
| **College** |  |  |  |  |
| Player ID | College ID | Zip Code | Name |  |
|  |  |  |  |  |
| **Conference** |  |  |  |  |
| Conference ID | Team ID | Name | Winner | Times Won |
|  |  |  |  |  |
| **Division** |  |  |  |  |
| Division ID | Team ID | Name | Season | Champion |

**3NF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Player** |  |  |  |  |
| Player ID | College ID | First Name | Last Name | Date of Birth |
|  |  |  |  |  |
| **Team** |  |  |  |  |
| Team ID | Division ID | Conference ID | Zip Code | Name |
|  |  |  |  |  |
| **Location** |  |  |  |  |
| Zip Code | State | City |  |  |
|  |  |  |  |  |
| **College** |  |  |  |  |
| Player ID | College ID | Zip Code | Name |  |
|  |  |  |  |  |
| **Conference** |  |  |  |  |
| Conference ID | Team ID | Name |  |  |
|  |  |  |  |  |
| **Division** |  |  |  |  |
| Division ID | Team ID | Name | Season | Champion |
|  |  |  |  |  |
| **Conference-Winner** |  |  |  |  |
| Winner | Times Won |  |  |  |

**5. Recommendation on DBMS and rationale**

MySQL WorkBench

**Flexible privilege and password system**-To allow certain permission for logins(Ex. Coaches, Owners,Stat and Record keepers.Flexible privilege and password system to make sure records and data are not easy falsified.

**Scalability-**the database can be small as 1MB or can change and grown to match the size of the league.

**Web and Data Warehouse-** power to move the database from standalone to web. When beginning have the data on a standalone machine, once a website is need so more people can see the data. Can have the database web based in the future pair with fantasy engines and globalized market.

**Strong Data Protection -** We are dealing with data that can affect large amounts of people. Teams making deals with players and vis versa need accurate and fast data to make decisions.

**Management Ease-**For beginners MySQL offers exceptional quick-start capability with the average time from software download to installation completion being less than fifteen minutes.

**Open Source Freedom and 24 \* 7 Support-**Great feature when you have a database that will need to be accessed on a 24 hour basis especially when paired with a website.

**6. Implementation-Ready Data Model**

For our database we are choosing to use the third normal form. By having our database normalized to the third normal form we are sacrificing possible performance, however we gain the advantage of having the data stored all in one place. We also obtain data that isn’t redundant and get an overall better design because of it. When transitioning our database to the 1NF we realized there were no attributes that were multivalued. However even know there were none existing in our own database it was important to still check them during the 1NF. After normalizing the database into the 2NF and finally 3NF each Primary Key is fully functionally dependent. This allows for a direct relation between the primary key and all of the attributes in the table. When implementing the 3NF ‘Conference Winners’ and the attribute ‘Times Won’ were given their own table. This is because the number of times won directly depends on who the Winner is. After normalizing to this level the ‘Conference Winners’ table has less of a chance of having errors when the data is updated and runs less of a risk of having data that is unsynchronized.

**7. Data Dictionary**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | **Attribute Name** | **Type** | **Format** | **Range** | **Required** | **PK or FK** | **Fk Reference Table** | **Content** |
| Player | player\_id | int(4) | 999999 | 000000-999999 | Y | PK |  | Player's unique id |
|  | DOB | date() | YYYY-MMM-DD |  | Y |  |  | Date player was born |
|  | f\_Name | vchar(45) | Xxxxxxx |  | Y |  |  | Player’s first name |
|  | l\_Name | vchar(45) | Xxxxxxx |  | Y |  |  | Players last name |
|  | college\_id | int(4) | 999999 | 000000-99999999 | Y | FK | College | Unique college id |
|  |  |  |  |  |  |  |  |  |
| College | college\_id | int(4) | 999999 | 000000-999999 | Y | PK |  | College’s unique id |
|  | name | vchar(45) | xxxxxx |  | Y |  |  | Name of the College |
|  | zipCode | int(9) | 99999999 | 000000-99999999 | Y | FK | Location | Unique id for zipCode table |
|  |  |  |  |  |  |  |  |  |
| Team |  |  |  |  |  |  |  |  |
|  | team\_id | int(4) | 999999 | 000000-999999 | Y | PK |  | Unique team id |
|  | zipcode | char(6) | 99999 | 00000-999999 |  | FK | Location | Unique key to reference zip code table |
|  | name | vchar(45) | xxxxxx |  | Y |  |  | Name of the team |
|  | division\_id | int(4) | 999999 | 000000-999999 | Y | FK | Team | Unique id for division table |
|  | conferenc\_id | int(4) | 999999 | 000000-999999 | Y | FK | Conference | Unique attribute for conference |
|  |  |  |  |  |  |  |  |  |
| Location | zipCode | char(6) | 99999 | 000000-999999 | Y | PK |  | Unique attribute for zipCode table |
|  | city | vchar(45) | xx |  | Y |  |  | City matching the zipcode |
|  | state | vchar(45) | xxxxxx |  | Y |  |  | State matching the city for the zip code |
|  |  |  |  |  |  |  |  |  |
| Division | division\_id | int(4) | 999999 | 000000-999999 | Y | PK |  | Unique attribute distinguish the divisions. |
|  | name | vchar(45) | xxxxx |  | Y |  |  | Name of division |
|  | season | vchar(45) | xxxxx |  |  |  |  | Season of division |
|  | champion | vchar(45) | xxxxx |  |  |  |  | Store name of the champion of t he Division |
| Conference | conference\_id | int(4) | 999999 | 0000000-999999 | Y | PK |  | Unique attribute for conference table |
|  | name | vchar(45) | xxxxxx |  | Y |  |  | Name of Conference |
|  | winner | vchar(45) | xxxxxx |  |  |  |  | Name of the winner of the conference |
|  | Times won | tinyint(1) | xxxxxx |  |  |  |  |  |

**8. SQL**

-- MySQL Script generated by MySQL Workbench

-- 12/06/16 21:20:36

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

-- -----------------------------------------------------

-- Schema NBA\_Players

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema NBA\_Players

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `NBA\_Players` ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`LOCATION`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`LOCATION` ;

CREATE TABLE IF NOT EXISTS `mydb`.`LOCATION` (

`Zip\_Code` INT(5) NOT NULL,

`State` VARCHAR(45) NULL,

`City` VARCHAR(45) NULL,

PRIMARY KEY (`Zip\_Code`))

ENGINE = InnoDB;

USE `NBA\_Players` ;

-- -----------------------------------------------------

-- Table `NBA\_Players`.`COLLEGE`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `NBA\_Players`.`COLLEGE` ;

CREATE TABLE IF NOT EXISTS `NBA\_Players`.`COLLEGE` (

`college\_id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

`zip\_code` INT NOT NULL,

PRIMARY KEY (`college\_id`, `zip\_code`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `NBA\_Players`.`PLAYER`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `NBA\_Players`.`PLAYER` ;

CREATE TABLE IF NOT EXISTS `NBA\_Players`.`PLAYER` (

`player\_id` INT NOT NULL AUTO\_INCREMENT,

`college\_id` INT NULL,

`dob` DATE NULL,

`f\_name` VARCHAR(45) NULL,

`l\_name` VARCHAR(45) NULL,

`COLLEGE\_college\_id` INT NOT NULL,

`COLLEGE\_zip\_code` INT NOT NULL,

PRIMARY KEY (`player\_id`),

INDEX `fk\_PLAYER\_COLLEGE\_idx` (`COLLEGE\_college\_id` ASC, `COLLEGE\_zip\_code` ASC),

CONSTRAINT `fk\_PLAYER\_COLLEGE`

FOREIGN KEY (`COLLEGE\_college\_id` , `COLLEGE\_zip\_code`)

REFERENCES `NBA\_Players`.`COLLEGE` (`college\_id` , `zip\_code`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `NBA\_Players`.`TEAM`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `NBA\_Players`.`TEAM` ;

CREATE TABLE IF NOT EXISTS `NBA\_Players`.`TEAM` (

`team\_id` INT NOT NULL AUTO\_INCREMENT,

`zip\_code` INT NOT NULL,

`division\_id` INT NOT NULL,

`conference\_id` INT NOT NULL,

`name` VARCHAR(45) NULL,

PRIMARY KEY (`team\_id`, `zip\_code`, `division\_id`, `conference\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `NBA\_Players`.`DIVISION`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `NBA\_Players`.`DIVISION` ;

CREATE TABLE IF NOT EXISTS `NBA\_Players`.`DIVISION` (

`division\_id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

`champion` VARCHAR(45) NULL,

`season` VARCHAR(45) NULL,

`TEAM\_team\_id` INT NOT NULL,

`TEAM\_zip\_code` INT NOT NULL,

`TEAM\_division\_id` INT NOT NULL,

`TEAM\_conference\_id` INT NOT NULL,

PRIMARY KEY (`division\_id`, `TEAM\_team\_id`, `TEAM\_zip\_code`, `TEAM\_division\_id`, `TEAM\_conference\_id`),

INDEX `fk\_DIVISION\_TEAM1\_idx` (`TEAM\_team\_id` ASC, `TEAM\_zip\_code` ASC, `TEAM\_division\_id` ASC, `TEAM\_conference\_id` ASC),

CONSTRAINT `fk\_DIVISION\_TEAM1`

FOREIGN KEY (`TEAM\_team\_id` , `TEAM\_zip\_code` , `TEAM\_division\_id` , `TEAM\_conference\_id`)

REFERENCES `NBA\_Players`.`TEAM` (`team\_id` , `zip\_code` , `division\_id` , `conference\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `NBA\_Players`.`LOCATION`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `NBA\_Players`.`LOCATION` ;

CREATE TABLE IF NOT EXISTS `NBA\_Players`.`LOCATION` (

`zip\_code` INT NOT NULL AUTO\_INCREMENT,

`state` VARCHAR(45) NULL,

`city` VARCHAR(45) NULL,

`COLLEGE\_college\_id` INT NOT NULL,

`COLLEGE\_zip\_code` INT NOT NULL,

`TEAM\_team\_id` INT NOT NULL,

`TEAM\_zip\_code` INT NOT NULL,

`TEAM\_division\_id` INT NOT NULL,

`TEAM\_conference\_id` INT NOT NULL,

PRIMARY KEY (`zip\_code`),

INDEX `fk\_LOCATION\_COLLEGE1\_idx` (`COLLEGE\_college\_id` ASC, `COLLEGE\_zip\_code` ASC),

INDEX `fk\_LOCATION\_TEAM1\_idx` (`TEAM\_team\_id` ASC, `TEAM\_zip\_code` ASC, `TEAM\_division\_id` ASC, `TEAM\_conference\_id` ASC),

CONSTRAINT `fk\_LOCATION\_COLLEGE1`

FOREIGN KEY (`COLLEGE\_college\_id` , `COLLEGE\_zip\_code`)

REFERENCES `NBA\_Players`.`COLLEGE` (`college\_id` , `zip\_code`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_LOCATION\_TEAM1`

FOREIGN KEY (`TEAM\_team\_id` , `TEAM\_zip\_code` , `TEAM\_division\_id` , `TEAM\_conference\_id`)

REFERENCES `NBA\_Players`.`TEAM` (`team\_id` , `zip\_code` , `division\_id` , `conference\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `NBA\_Players`.`CONFERENCE`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `NBA\_Players`.`CONFERENCE` ;

CREATE TABLE IF NOT EXISTS `NBA\_Players`.`CONFERENCE` (

`conference\_id` INT NOT NULL AUTO\_INCREMENT,

`winner` VARCHAR(45) NULL,

`times\_won` TINYINT(1) NULL,

`name` VARCHAR(45) NULL,

`DIVISION\_division\_id` INT NOT NULL,

PRIMARY KEY (`conference\_id`, `DIVISION\_division\_id`),

INDEX `fk\_CONFERENCE\_DIVISION1\_idx` (`DIVISION\_division\_id` ASC),

CONSTRAINT `fk\_CONFERENCE\_DIVISION1`

FOREIGN KEY (`DIVISION\_division\_id`)

REFERENCES `NBA\_Players`.`DIVISION` (`division\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

**Useful Queries**

List names of players that went to USER\_INPUT college name:

SELECT NAME

FROM PLAYER

LEFT JOIN COLLEGE ON PLAYER.college\_id = COLLEGE.college\_id

WHERE COLLEGE.name LIKE ‘USER INPUT’;

A fan wants to find Kobe Bryant’s birthday:  
  
SELECT DOB

FROM PLAYER

WHERE f\_name = 'Kobe' AND l\_name = 'Bryant';

Display team information about every team that won their division:  
  
SELECT \* FROM TEAM

INNER JOIN DIVISION ON DIVISION.champion = TEAM.name;

Print the city and the college of all NBA stars:

SELECT f\_name, l\_name, COLLEGE.name, city

FROM PLAYER

INNER JOIN COLLEGE on COLLEGE.college\_id = PLAYER.player\_id

INNER JOIN LOCATION on COLLEGE.zip\_code = LOCATION.zip\_code;

A coach wants to know number of players that played college and are in the NBA:

SELECT COUNT(DISTINCT PLAYER\_ID)

FROM PLAYER

WHERE COLLEGE\_ID IS NOT NULL;

A coach wants a list of players over 30:

SELECT F\_NAME, L\_NAME

FROM PLAYER

WHERE DOB < ‘1986-12-06’;

Print the season, the times they won, name of the conference and the winner’s name:

SELECT WINNER, TIMES\_WON, SEASON, CONFERENCE.NAME

FROM CONFERENCE, DIVISION

WHERE CONFERENCE.WINNER = DIVISION.CHAMPION;

Print all division winners by season in an ascending manner:

SELECT CHAMPION, SEASON

FROM DIVISION

GROUP BY SEASON

ORDER BY SEASON ASC;

Print the conference name and division name of each team:

SELECT TEAM.NAME, CONFERENCE.NAME, DIVISION.NAME

FROM TEAM, CONFERENCE, DIVISION

WHERE TEAM.CONFERENCE\_ID = CONFERENCE.CONFERENCE\_ID AND TEAM.DIVISION\_ID = DIVISION.DIVISION\_ID;

Fan wants to know which state the most players originated from:

SELECT F\_NAME, L\_NAME, STATE

FROM PLAYER, COLLEGE, LOCATION

WHERE PLAYER.COLLEGE\_ID = COLLEGE.COLLEGE.ID AND COLLEGE.ZIP\_CODE = LOCATION.ZIP\_CODE

GROUP BY STATE

ORDER BY(\*) ASC;

Fan wants to know which state and team has the most wins:

SELECT TEAM.NAME, LOCATION.STATE, MAX(CONFERENCE.TIMES\_WON) AS WON\_THE\_MOST

FROM TEAM

INNER JOIN CONFERENCE ON CONFERENCE.CONFERENCE\_ID = TEAM.CONFERENCE\_ID

INNER JOIN LOCATION ON LOCATION.ZIP\_CODE = TEAM.ZIP\_CODE;

**9. Time Log**

|  |  |  |
| --- | --- | --- |
| **Task** | **Hours Spent** | **Team Member** |
| **System Requirements** | **1.5** | **Victoria** |
| **Contextual Data Flow Diagram** | **2.0** | **Victoria** |
| **ER/EER** | **4.0** | **Victoria** |
| **Recommendation on DBMS and Rationale** | **1.0** | **Jarrod** |
| **Normalization Model** | **3.0** | **Stallone** |
| **Data Dictionary** | **2.0** | **Jarrod, Stallone** |
| **SQL** | **3.0** | **Jarrod, Victoria** |
| **Useful Queries** | **2.0** | **Jarrod, Stallone** |
| **Time Logs** | **0.5** | **Jarrod, Stallone, Victoria** |
| **Hair Pulling** | **10.0** | **Jarrod, Stallone, Victoria** |

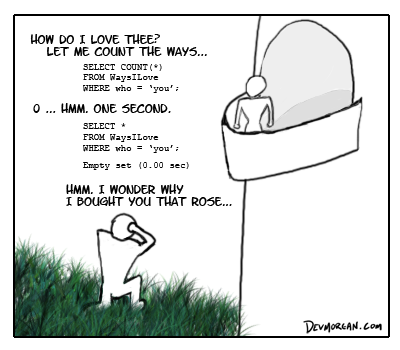
**Signatures**

Stallone Eu:

Victoria Green:

Jarrod Vega:

**10. Appendix**

****